This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method of processing memory reads and writes in a packet processor comprising the steps of:

receiving a ~~processing~~ memory read ~~reads~~ having a first ~~an~~ associated sequence number;

receiving a ~~processing~~ memory write ~~writes~~ having a second ~~an~~ associated sequence number;

determining if a memory conflict exists between said memory read and said memory write based on a comparison of said first associated sequence number and said second associated sequence number; and

signaling a restart based on said step of determining, utilizing said first associated ~~of a particular~~ sequence number ~~numbers when a memory conflict is detected~~.

2. (Currently Amended) The method of Claim 1 wherein said ~~the~~ step of receiving a ~~processing~~ memory read ~~reads~~ comprises the steps of:

recording a first ~~the~~ address and said first associated sequence number in a read table;

searching a write table for an entry with an ~~a matching~~ address matching said first address and a sequence number equal to or less than said first associated sequence number ~~the received read~~;

if said step of searching succeeds, ~~upon detecting a match,~~ retrieving data from said ~~the~~ write table.

3. (Currently Amended) The method of Claim 2 wherein said ~~the~~ step of receiving

a ~~processing~~ memory write ~~writes~~ comprises the step ~~steps~~ of:

recording a second ~~the write~~ address, said second associated ~~write~~ sequence number and write data in said ~~a~~ write table;

and wherein said step of determining if a memory conflict exists comprises the step of:

searching said ~~a~~ read table for an entry with an ~~a matching~~ address matching said second address and a sequence number greater than said second associated sequence number ~~the received memory write~~.

4. (Currently Amended) The method of Claim 3 further comprising the step of:

receiving ~~processing~~ a start signal with a third ~~an~~ associated sequence number; and

receiving a done signal with a fourth ~~an~~ associated sequence number;

wherein ~~to maintain~~ a list of active sequence numbers is maintained.

5. (Currently Amended) The method of Claim 4 further comprising the steps of:

sending write data to a memory system upon receipt of said done signal;

flushing the entries in said read table and said write table corresponding to said fourth associated sequence number ~~associated with said done signal~~.

In the Specification, please add the following paragraph after paragraph 0021 and before paragraph 0022:

> [0021a] In the present specification the term "virtual sequentiality" is used to refer to the fact that sequential processing of packets is simulated but does not actually occur. That is, results are guaranteed to be consistent with sequential processing of each packet one at a time, while in fact packets may be processed in parallel and possibly out of sequence.

In the Specification, please change paragraph 0044 (the Abstract) as follows:

> [0044] A mechanism processes memory reads and writes in a packet processor. Each memory access has an associated sequence number and information is maintained allowing the detection of memory ~~ordering~~ conflicts. The mechanism is placed between a processing element and a memory system such that write data is buffered and both reads and writes are recorded. When a memory ~~an ordering~~ conflict is detected, based on a strict or alternate ordering model, a restart signal is generated and the entries for the associated sequence number are flushed. When the work associated with a sequence number has completed, a signal is made so that associated write data can be sent to the memory system and the entries for that sequence number can be flushed.

In the Specification, please change paragraph 0008 as follows:

> [0008] In these cases where simultaneous processing of packets is required, and where dependencies can exist between packets, it can be complicated to enforce those dependencies. Currently, there are two common approaches to this problem. The first solution is a software solution, where software locks are included in the code to cause dependent packet processing to be delayed until an earlier packet has been completed. These software semaphores are used to lock out subsequent dependent packets from accessing state until the first packet has updated it. The second solution involves hardware, where packet classification hardware serializes all packets that can possibly be dependent. In a multiprocessor, this can involve generating a hash function that sends all packets of the same flow to the same processor, and distributes the load across multiple processors.